UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/037,806	12/26/2001	Thomas J. Bonola	1662-49800 JMH (P98-2417)	6235

22879 7590 04/28/2005

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
INOA, MIDYS	

ART UNIT	PAPER NUMBER
2189	

DATE MAILED: 04/28/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/037,806	BONOLA, THOMAS J.	
	Examiner	Art Unit	
	Midys Inoa	2189	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *07 March 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,3-30 and 32-38* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,3-30 and 32-38* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *26 December 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Response to Arguments

1. Applicant's arguments filed on March 7th, 2005 have been fully considered but they are not persuasive.

Applicant argues that Trainin does not make a distinction between returning free memory blocks by a software stream or a hardware device.

Training discloses allocating and de-allocating memory blocks from a linked list to be used in tasks being run in hardware devices. The allocation process described by Trainin (paragraphs 0039 – 0040) depicts the execution of software steps that complete the allocation process (performing by a software stream...). However, although it is clear that these steps are performed in software, it is understood that software resides in hardware and thus, hardware is also involved in the allocation process.

Likewise, Trainin discloses a de-allocation process (paragraph 0041-0042) depicted by the execution of steps. Although Trainin does not distinguish the de-allocation process as being performed by software or hardware, de-allocation occurs when the hardware is no longer using the memory and the de-allocation process is performed with the use of instructions within the hardware. Therefore, both hardware and software are involved in the de-allocation process.

Using this rationale, the Examiner would like to explain that since the allocation and de-allocation processes involve both hardware and software; the system of Trainin does teach performing memory block allocation by a software stream and returning memory blocks by a hardware device.

Additionally, the Examiner would like to clarify that the claims themselves as presented do not demonstrate a distinction between performing memory operations by a software stream and returning memory blocks by a hardware device. Applicant does not explain how performing an operation via software does not involve hardware or how performing an operation via hardware does not involve software. It is understood that in performing software operations, the hardware housing the software must be involved and in performing allocation operations via hardware, allocation algorithms must be involved.

2. Applicant's arguments regarding claims 1, 3-11, 29-30, and 32-38 have been considered but are moot in view of new grounds of rejection.

Claim Rejections - 35 USC § 112

3. Claims 1, 12, 19, and 29 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. It is not clear how software can perform memory operations without involving any hardware and it is not clear how hardware can perform the return of memory blocks without the use of software.

Claim Rejections - 35 USC § 102

4. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –

> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

5. Claims 12-28 are rejected under 35 U.S.C. 102(e) as being anticipated by Trainin et al. (US 2002/0144073).

Regarding Claim 12, Trainin discloses a method of managing a heap memory comprising:

maintaining unused blocks of heap memory as a linked list, and wherein the unused blocks of the linked list comprise a first block at a beginning of the linked list, a second block pointed to the first block, and a third block at an end of the linked list (Figure 4);

removing, by a software stream, the first block from the linked list, thus making the second block the beginning of the linked list ("memory block from the heap is allocated and removed from the linked list of free blocks", Page 3, Paragraph 36);

and returning a return block, by a hardware device that used the return block, to the linked list by placing the return block at the end of the linked list (freed blocks are returned to the linked list, Page 3, Paragraph 38), wherein blocks are allocated to tasks being performed by a hardware device using the memory (Page 3, paragraph 0032) and newly freed blocks are freed by the hardware device that was using them once they are no longer in use.

In freeing these blocks, the return to the linked list is initiated; therefore, the hardware device that was using the blocks initiates the return to the linked list by freeing the blocks. Additionally, in the allocation and de-allocation process, memory blocks are allocated to tasks (software) being run in a processor (hardware); therefore, the allocation (paragraph 039) and the de-allocation (paragraph 0041) process involve both hardware and software.

Regarding Claim 13, Trainin discloses a method for placing a freed block of a heap memory at the top of a linked list (first end of the linked list) and modifying the same method to place the block of heap memory at the bottom of the linked list (second end of the linked list). To perform this task, a "null" pointer must, be written in the next block field of the return block of the heap memory; the block number of the return block of heap memory must be writing in the next block field of a last block of heap memory in the linked list; and the contents of the bottom register must be changed to point to the return block of heap memory. These changes allow for the new block of heap memory to be properly placed as the last entry of the linked list (page 3, paragraph 38).

Regarding Claims 14-15, Trainin discloses the method of allocating blocks from a heap memory and removing such block from the linked lists by: determining a block number of the primary block; reading a next block field of the primary block of memory ("next free block address"); and removing the primary block if the next block field of the primary block does not indicate a null (Page 3, paragraph 37). This system checks if the next block field of the primary block is a "null", indicating that this is the last block of the linked list, prior to removing a block of heap memory from the linked list. Additionally, when removing the top block of the heap memory, the new top block must be adjusted to point to the top of the heap ("writing the block number of the second block to the top register"). In determining the block number of the primary block, this block number must be read from the top register in the linked list.

Regarding Claims 16-18, Trainin discloses a method for placing a freed (fourth) block of a heap memory at the top of a linked list (beginning of the linked list). To accomplish this, the system has to read a top register, the top register identifying the beginning of the linked list

("address of the first free block"), write the block number the block identified by the top register to a next block field of the freed block of heap memory ("next free block address is updated to point to the former first free block"); and write the block number of the freed block to a top register. These changes allow for the new block of heap memory (fourth block) to be properly placed as the first entry of the linked list (page 3, paragraph 38).

Regarding Claim 19, Trainin discloses a method of managing a heap memory in a computer system, the method comprising:

allowing a software thread to add and remove blocks of heap memory from a linked list of free blocks of heap memory in a last-in/first-out fashion at a first end of the linked list (memory blocks allocated for use and removed from the linked list... when freed, are returned to the linked list, Page 3, paragraph 32-38) wherein allocating the memory blocks for use by a hardware device, the system is essentially using a software stream since the hardware device in question (processor) is driven by software (tasks);

and allowing a hardware device that uses blocks of heap memory to add blocks the of heap memory to the linked list of free blocks of heap memory at a second end of the linked list, wherein blocks are allocated to tasks (software) being performed by a processor (hardware device) using the memory (Page 3, paragraph 0032) and newly freed blocks are freed by the hardware device that was using them once they are no longer in use. In freeing these blocks, the return to the linked list is initiated; therefore, the hardware device that was using the blocks initiates the return to the linked list.

Trainin discloses the ability to return and remove blocks from both ends of the linked list. When performing both removal and return operations at the same end of the linked list, the

linked list is behaving in a LIFO fashion. It is understood that in the process of returning and removing blocks from the linked list, both hardware (such as buses for transfer of data and registers for holding data) and software (such as allocation software) are used.

In the allocation and de-allocation process, memory blocks are allocated to tasks (software) being run in a processor (hardware); therefore, the allocation (paragraph 039) and the de-allocation (paragraph 0041) process involve both hardware and software.

Regarding Claims 20-21, Trainin discloses the method of managing a heap memory in a computer system wherein allowing a software thread to remove blocks of heap memory in LIFO fashion further comprises: determining, by the software thread, a block number of a block of heap memory at the first end of the linked list ("first free block"), and removing the block of heap memory at the first end of the linked list (Page 3, paragraph 39). In determining the block number of the first free block of the linked lists, this number is read from the linked list (Paragraphs 32-3 9).

Regarding Claim 22, Trainin discloses a method of managing a heap memory in a computer system wherein removing the block of heap memory at the first end of the linked list further comprises: reading a next block field of the block of heap memory at the first end of the linked list to identify a block number of a next block in the linked list ("next free block address"); and writing the block number of the next block in the linked list to the beginning register (Page 3-4, Paragraphs 32-42).

Regarding Claim 23-25, Trainin discloses a method of managing a heap memory in a computer system wherein allowing a software thread to add blocks of heap memory in LIFO fashion further comprises: determining, by the software thread, a block number of a block of

heap memory at the first end of the linked list ("first free block of the linked list"); writing the block number of the block of heap memory at the first end of the linked list to a next block field of a return block of heap memory ("the next free block address is updated to point to the former first free block") and making the return block of heap memory the first end of the linked list. These changes allow for the new block of heap memory to be properly placed as the first entry of the linked list (page 3, paragraph 38). In determining the block number of the first free block of the linked list, this data must be read from the linked list.

Regarding Claims 26-28, Trainin discloses a method for placing a freed block of heap memory at the beginning of a linked list. In addition, Trainin discloses that the same method, only modified can be used to place the block of heap memory at the end of the linked list. In reversing this method, the system must: determine, by reading, a block number of a block of heap memory the second end of the linked list; write a block number of a return block of heap memory to a next block field of the block of heap memory at the second end of the linked list; and making the return block of heap memory the second end of the linked list (Page 3, Paragraph 38).

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1, 3-11, 29-30, and 32-38 are rejected under 35 U.S.C. 103(a) as being unpatentable over Trainin et al. (US 2002/0144073) in view of Steele, JR et al. (2001/0056420).

Regarding Claims 1 and 7, Trainin et al. discloses:

performing, by a software stream, heap memory operations on a first end of a linked list of free heap memory of a heap pile ("block from heap is allocated for use and removed from the linked list...") wherein the memory blocks are allocated to tasks (software) being run by a processor (a hardware device), thus the system is essentially using a software stream for allocation since hardware devices are driven by software;

and returning, by a hardware device that used the return block, a return block of heap memory to the heap pile at a second end of the linked list of free heap memory (when no longer needed, it is freed and returned to the linked list), wherein blocks are allocated to tasks (software) being performed by a hardware device using the memory (Page 3, paragraph 0032) and newly freed blocks are freed by the hardware device that was using them once they are no longer in use;

in freeing these blocks, the return to the linked list is initiated, therefore, the hardware device that was using the blocks initiates the return to the linked list by feeing the blocks.

In this system, allocation software is used to allocate the blocks that are being removed from the linked list. Hardware components are also used in the transferring of the data in the blocks and in the storage of the data. In this case, the first end of the heap is the bottom of the heap, and the second end is the beginning of the heap (Page 3, paragraphs 36- 38). Additionally, in the allocation and de-allocation process, memory blocks are allocated to tasks (software) being run in a processor (hardware); therefore, the allocation (paragraph 039) and the de-allocation (paragraph 0041) process involve both hardware and software.

Trainin et al. does not teach concurrently returning a block of heap memory to the heap pile while heap operations are being performed on the other end of the linked list. Steele, JR. et al. discloses improving memory access speeds by performing concurrent push and pop operations at both ends of the double ended queue (Page 2, paragraph 18). It would have been obvious to one of ordinary skill in the art at the time the invention was made to adjust the invention of Trainin et al, so that the allocation (popping of memory blocks from the queue) and returning (pushing of memory blocks in to the queue) of blocks to the heap are done concurrently since doing so would save processing time and make the system more versatile.

Regarding Claim 3, Trainin discloses a method for placing a freed block of a heap memory at the top of a linked list (first end of the linked list) and modifying the same method to place the block of heap memory at the bottom of the linked list (second end of the linked list). To perform this task, a "null" pointer must be written in the next block field of the return block of the heap memory; the block number of the return block of heap memory must be writing in the next block field of a last block of heap memory in the linked list; and the contents of the bottom register must be changed to point to the return block of heap memory. These changes allow for the new block of heap memory to be properly placed as the last entry of the linked list (page 3, paragraph 38).

Regarding Claim 4, Trainin discloses a method for placing a freed block of a heap memory at the top of a linked list (first end of the linked list) and modifying the same method to place the block of heap memory at the bottom of the linked list (second end of the linked list). This method can be used to return block of heap memory and place it at the first end (top) of the

linked list. In placing this block at the top of this linked list, this block will be the first block available for use (Page 3, paragraph 38).

Regarding Claim 5, Trainin discloses a method for placing a freed block of a heap memory at the top of a linked list (first end of the linked list). To accomplish this, the system has to determine the block number of a primary block of heap memory resident at the first end of the linked list (top of the linked list.. ."address of the first free block") write the block number of the primary block of heap memory to a next block field of the freed block of heap memory ("next free block address is updated to point to the former first free block"); and write the block number of the freed block to a top register. These changes allow for the new block of heap memory to be properly placed as the first entry of the linked list (page 3, paragraph 38).

Regarding Claim 8, Trainin discloses a method for placing a freed block of a heap memory at the top of a linked list (first end of the linked list), modifying the same method to place the block of heap memory at the bottom of the linked list (second end of the linked list) and another method for allocating blocks from the heap for use and removing them from the linked lists. Trainin's allocation and removal method can also be modified to remove blocks from the bottom of the linked lists. Therefore, Trainin discloses the method of removing heap memory from the linked list heap management system by taking a primary block of heap memory resident at the first end of the of the linked list (Paragraphs 3 6-40).

Regarding Claims 6 and 10, Trainin discloses setting the new block's "next free block address" to that of the primary block's address; in order to do this, the address of the primary block (former first block) must be read by the system (Page 3, paragraph 38). In determining the

block number of a primary block (former first block) of heap memory resident at the first end of the linked list, the top register is read prior to writing the block number of the second block.

Regarding Claims 9 and 11, Trainin discloses the method of allocating blocks from a heap memory and removing such block from the linked lists by: determining a block number of the primary block; reading a next block field of the primary block of memory ("next free block address"); and removing the primary block if the next block field of the primary block does not indicate a null (Page 3, paragraph 37). This system checks if the next block field of the primary block is a "null", indicating that this is the last block of the linked list, prior to removing a block of heap memory from the linked list. Additionally, when removing the top block of the heap memory, the new top block must be adjusted to point to the top of the heap ("writing the block number of the second block to the top register"). In determining the block number of the primary block, this block number must be read from the top register in the linked list.

Regarding Claims 29 and 32, Trainin discloses a computer system comprising:

a microprocessor executing a software stream (computer system for memory allocating);

a main memory array, a portion of the main memory array allocated to be a heap memory, and wherein unused portions of the heap memory are part of a heap pile, the heap pile further comprising (Figure 4 and Page 3, paragraphs 32-36) a plurality of blocks (see Figure 4);

each block having a next block field ("next free block address");

wherein the heap pile is maintained as a linked list, each block's next block field pointing to a next block in the list (paragraphs 36-3 8);

a first bridge logic device coupling the microprocessor to the main memory array;

a hardware device coupled to the heap memory through the first bridge logic device;

wherein the software stream executed on the microprocessor removes blocks of heap memory from a beginning of the heap pile (allocated blocks are removed from the linked list, paragraph 36) and in allocating the memory blocks for use by a hardware device, the system is essentially using a software stream since the hardware device is driven by software;

and the hardware device returns blocks of heap memory to an end of the heap pile (freed blocks are returned to the linked list, paragraph 38), wherein blocks are allocated to tasks being performed by a hardware device using the memory (Page 3, paragraph 0032) and newly freed blocks are freed by the hardware device that was using them once they are no longer in use; in freeing these blocks, the return to the linked list is initiated, therefore, the hardware device that was using the blocks initiates the return to the linked list by freeing the blocks.

In this system allocation software is used to allocate the blocks that are being removed from the linked list, Hardware components are also used in the transferring of the data in the blocks; such as buses and registers. Additionally, in the allocation and de-allocation process, memory blocks are allocated to tasks (software) being run in a processor (hardware); therefore, the allocation (paragraph 039) and the de-allocation (paragraph 0041) process involve both hardware and software. In this case, the first end of the heap is the bottom of the heap, and the second end is the beginning of the heap (Page 3, paragraphs 36-38).

Trainin et al. does not teach simultaneously returning a block of heap memory to the heap pile while heap operations are being performed on the other end of the linked list. Steele, JR. et al. discloses improving memory access speeds by performing concurrent push and pop operations at both ends of the double ended queue (Page 2, paragraph 18). It would have been obvious to one of ordinary skill in the art at the time the invention was made to adjust the

invention of Trainin et al, so that the allocation (popping of memory blocks from the queue) and returning (pushing of memory blocks in to the queue) of blocks to the heap are done concurrently since doing so would save processing time and make the system more versatile.

Regarding Claim 30, Trainin discloses the computer system wherein plurality of blocks can each have the same number of bytes or could be combined to form larger blocks of heap memory (Page 3, Paragraph 39).

Regarding Claims 33-38, Trainin discloses allocating memory blocks in the heap for use in any task involving any of many hardware devices within a computer system. Therefore, the hardware device in question could be a graphics card, a network card, an audio card, a hard drive, or any other storage device or computer component (Page 3, paragraph 33).

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Midys Inoa whose telephone number is (571) 272-4207. The examiner can normally be reached on M-F 5:30am - 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Mano Padmanabhan can be reached on (571) 272-4210. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

April 27, 2005

Midys Inoa
Examiner
Art Unit 2189

MI

Pierre Vital
Primary Examiner
Art Unit 2188